JOINS και AGGREGATES
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Αντώνιος Δεληγιαννάκης

Θυμηθείτε …

Θα χρησιμοποιούμε ως παράδειγμα έναν πίνακα Student με το ακόλουθο Σχήμα

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

sID: Student ID / Αναγνωριστικό Μαθητή

sName: Όνομα Μαθητή

GPA: Μέσος όρος (βαθμολογίας – μέγιστη τιμή το 4.0)

hs: Πληθυσμός σχολείου στο οποίο φοιτά

Δύο Νέοι Πίνακες

Θα χρησιμοποιούμε επίσης ως παράδειγμα έναν πίνακα College με το ακόλουθο Σχήμα

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

cName: Όνομα Κολλεγίου

state: Πολιτεία στην οποία βρίσκεται το κολλέγιο

enr: Αριθμός φοιτητών στο κολλέγιο

Δύο Νέοι Πίνακες

Θα καταγράφουμε τις αιτήσεις των μαθητών σε Τμήματα των Κολλεγίων. Θα χρησιμοποιούμε ως παράδειγμα έναν πίνακα Apply με το ακόλουθο Σχήμα

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

sID: Student ID / Αναγνωριστικό Μαθητή

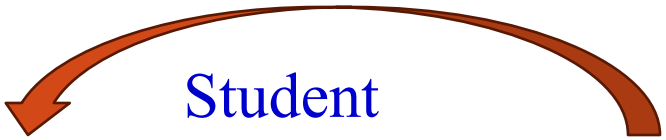
cName: Όνομα Κολλεγίου

major: Τμήμα/Σχολή του Κολλεγίου

dec: Αποτέλεσμα αίτησης (YES/NO/NULL)

Foreign Key / Ξένο Κλειδί

- Δείτε τα γνωρίσματα sID και cName του πίνακα Apply
- Παρατηρήστε ότι στο sID δε μπορεί να μπει οποιαδήποτε τιμή, αλλά μόνο τιμές που **πρέπει να υπάρχουν** στο πεδίο sID του πίνακα Student
- Όμοια, στο cName δε μπορεί να μπει οποιαδήποτε τιμή, αλλά μόνο τιμές που **πρέπει να υπάρχουν** στο πεδίο cName του πίνακα Apply



Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

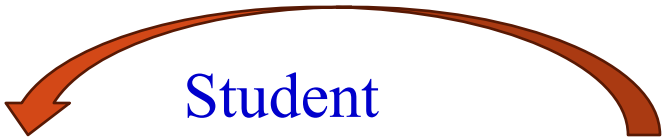
cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Foreign Key / Ξένο Κλειδί

- Περιορισμός Ξένου Κλειδιού (Foreign Key)
  - Δηλώνεται στο CREATE TABLE
  - Ένα ή περισσότερα attributes **αναφέρονται** στο κλειδί μιας άλλη σχέσης
  - Οι τιμές τους πρέπει να υπάρχουν στο κλειδί (συνήθως, στο PRIMARY KEY) μιας άλλης σχέσης
  - Τα attributes ενός foreign key μπορεί να επιτρέπεται να έχουν NULL τιμή. Αν όχι, πρέπει να τα δηλώσουμε ως NOT NULL
  - Το ΣΔΒΔ διασφαλίζει ότι ικανοποιούνται πάντα οι περιορισμοί αναφορικής ακεραιότητας (referential integrity constraint / foreign key)

CREATE TABLE + Foreign Key

```
CREATE TABLE Apply (
        sID     INT references Student(sID),
        cName VARCHAR(20) references College(cName),
        major   VARCHAR(20),
        dec     CHAR(3) NULL,
        PRIMARY KEY(sID, cName, major)
)
```



Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

CREATE TABLE + Foreign Key

```
CREATE TABLE Apply (
        sID     INT references Student(sID),
        cName VARCHAR(20) references College(cName),
        major   VARCHAR(20),
        dec     CHAR(3) NULL,
        PRIMARY KEY(sID, cName, major)
)
```

Αν έχουμε Foreign Key στο Primary Key μιας σχέσης, τα attributes του Primary Key δεν απαιτούνται στο references.

- `sID INT references Student,`
- `cName VARCHAR(20) references College,`

Αν το Foreign Key είναι σύνθετο (γιατί το κλειδί στο οποίο αναφερόμαστε είναι σύνθετο), τότε μπαίνει σε ξεχωριστή γραμμή

- `FOREIGN KEY(A, B) references OtherTable(Key1, Key2)`

Foreign Keys – Πώς Προκύπτουν;

- **Ποτέ** οι πίνακες και τα Foreign Keys δε μπαίνουν αυθαίρετα
- Πρώτα συλλέγουμε τις απαιτήσεις μιας εφαρμογής
  - Ποιες έννοιες αφορά;
  - Ποιοι θα τη χρησιμοποιούν;
  - Τι ερωτήματα/χρήση απαιτείται;
- Μετά σχεδιάζουμε το Λογικό Μοντέλο (ER ή UML διάγραμμα)
- Μετά μετατρέπουμε το ER ή UML διάγραμμα στο σχεσιακό σύστημα
  - Σε αυτό το βήμα προκύπτουν οι πίνακες της βάσης
  - Σε αυτό το βήμα προκύπτουν τα ξένα κλειδιά.

Joins + Foreign Keys

- Τα Foreign Keys δηλώνουν κάποιες σχέσεις μεταξύ πινάκων
- Στα Λογικά Μοντέλα οι σχέσεις μεταξύ των βασικών οντοτήτων/εννοιών αποτυπώνονται
  - Ως relationships στο ER διάγραμμα
  - Ως associations στο UML διάγραμμα

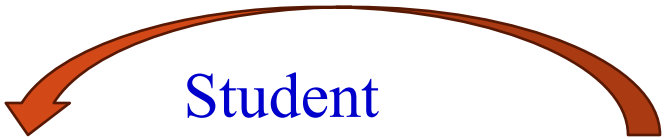
Joins + Foreign Keys

- Θα δούμε ότι κάθε relationship/association όταν κάνουμε τη μετατροπή στο σχεσιακό σύστημα οδηγεί στον ορισμό Foreign Key
  - Μέσω αυτού μπορεί να εκφραστεί η συσχέτιση και να συνδυαστούν τα δεδομένα 2 πινάκων που σχετίζονται κάπως
  - Foreign Keys μπορούν να δημιουργηθούν και από άλλου είδους σχέσεις, πχ από σχέσεις κληρονομικότητας

- Τα Joins μας επιτρέπουν να συνδυάσουμε δεδομένα διαφορετικών πινάκων
- Το JOIN δύο πινάκων δεν απαιτεί την ύπαρξη ξένου κλειδιού
  - Όμως >95% των JOINs γίνονται μεταξύ ενός ξένου κλειδιού ενός πίνακα με το πρωτεύον κλειδί στο οποίο αναφέρεται

Inner Join (ή Join)

- Συνδυάζει πλειάδες 2 σχέσεων υπό συνθήκη
- "Τύπωσε το αναγνωριστικό (sID) και το μέσο όρο (GPA) κάθε φοιτητή που έχει κάνει 1 ή περισσότερες αιτήσεις.
- Παρατηρήστε ότι τα attributes που θέλουμε να επιστραφούν (sID, GPA) βρίσκονται στον πίνακα Student, όμως οι φοιτητές που έχουν κάνει αίτηση βρίσκονται στον πίνακα Apply.
- **Αυτό σημαίνει ότι χρειαζόμαστε κάποιο Join**.
- Ένας φοιτητής που έχει κάνει πολλές αιτήσεις θέλουμε να εμφανιστεί 1 μόνο φορά => αφαίρεση διπλότυπων



Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

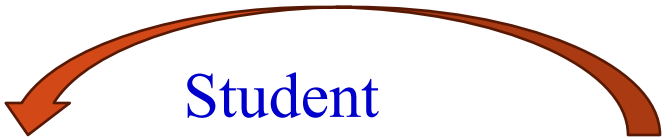
cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Inner Join (ή Join)

"Τύπωσε το αναγνωριστικό (sID) και το μέσο όρο (GPA) κάθε φοιτητή που έχει κάνει 1 ή περισσότερες αιτήσεις.

Προαιρετική η παρένθεση

```
SELECT DISTINCT Student.sID, GPA
FROM    Student JOIN Apply ON (Student.sID = Apply.sID)
```



Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Inner Join (ή Join)

Προαιρετική η παρένθεση

```
SELECT DISTINCT Student.sID, GPA
FROM    Student JOIN Apply ON (Student.sID = Apply.sID)
```

- Αντί για JOIN θα μπορούσαμε να είχαμε γράψει INNER JOIN
- Προσέξτε ότι επειδή το sID υπάρχει σε 2 σχέσεις που κάνουμε JOIN, πρέπει να μπορεί η βάση να καταλάβει σε ποιο από τα 2 αναφερόμαστε. Χρειάζεται να βάλουμε ως πρόθεμα (Student.sID και Apply.sID) το όνομα της σχέσης στην οποία ανήκουν.

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Inner Join (ή Join)

SELECT DISTINCT Student.sID, GPA

FROM Student JOIN Apply ON (Student.sID = Apply.sID)

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση
- Το Inner Join 2 σχέσεων επιστρέφει μία προσωρινή σχέση με τόσα γνωρίσματα όσο η ένωση των γνωρισμάτων των 2 σχέσεων
- Άρα, **πριν το SELECT DISTINCT**, το JOIN επιστρέφει έναν πίνακα με τις εξής στήλες:

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

SELECT DISTINCT Student.sID, GPA

FROM Student JOIN Apply ON (Student.sID = Apply.sID)

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT DISTINCT Student.sID, GPA
FROM    Student JOIN Apply ON (Student.sID = Apply.sID)
```

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT DISTINCT Student.sID, GPA
FROM    Student JOIN Apply ON (Student.sID = Apply.sID)
```

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

SELECT DISTINCT Student.sID, GPA

FROM Student JOIN Apply ON (Student.sID = Apply.sID)

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

SELECT DISTINCT Student.sID, GPA

FROM Student JOIN Apply ON (Student.sID = Apply.sID)

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

SELECT DISTINCT Student.sID, GPA

FROM Student JOIN Apply ON (Student.sID = Apply.sID)

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

SELECT DISTINCT Student.sID, GPA

FROM Student JOIN Apply ON (Student.sID = Apply.sID)

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT DISTINCT Student.sID, GPA
FROM    Student JOIN Apply ON (Student.sID = Apply.sID)
```

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT DISTINCT Student.sID, GPA
FROM    Student JOIN Apply ON (Student.sID = Apply.sID)
```

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO
3	Mary	3.5	1200	3	EMP	ECE	YES

Ισχύει ότι Student.sID = Apply.sID ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

SELECT DISTINCT Student.sID, GPA

FROM Student JOIN Apply ON (Student.sID = Apply.sID)

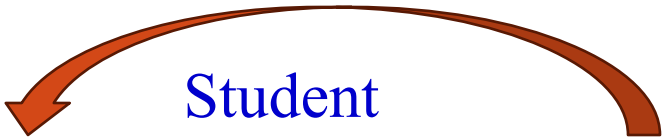
- Μετά εκτελείται το SELECT DISTINCT

Student.sID	GPA
1	3.0
3	3.5

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO
3	Mary	3.5	1200	3	EMP	ECE	YES

Άλλο Παράδειγμα

- Π.χ.: "Τύπωσε το όνομα (sName) κάθε μαθητή που έχει γίνει δεκτός (dec='YES') σε κάποιο τμήμα Κολλεγίου, μαζί το όνομα του Κολλεγίου (cName) και το όνομα του τμήματος (major).
- Παρατηρήστε ότι τα attributes που θέλουμε να επιστραφούν (sName, cName, major) και το attribute στο οποίο εκφράζουμε συνθήκη (dec='YES') ανήκουν σε διαφορετικούς πίνακες. **Αυτό σημαίνει ότι χρειαζόμαστε κάποιο Join**.



Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Inner Join (ή Join)

Προαιρετική η παρένθεση

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Αντί για JOIN θα μπορούσαμε να είχαμε γράψει INNER JOIN
- Προσέξτε ότι επειδή το sID υπάρχει σε 2 σχέσεις που κάνουμε JOIN, πρέπει να μπορεί η βάση να καταλάβει σε ποιο από τα 2 αναφερόμαστε. Χρειάζεται να βάλουμε ως πρόθεμα (Student.sID και Apply.sID) το όνομα της σχέσης στην οποία ανήκουν.

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM    Student JOIN Apply ON
            (Student.sID = Apply.sID AND dec = 'YES')
```

- Το Inner Join 2 σχέσεων επιστρέφει μία προσωρινή σχέση με τόσα γνωρίσματα όσο η ένωση των γνωρισμάτων των 2 σχέσεων
- Άρα, **πριν το SELECT**, το JOIN επιστρέφει έναν πίνακα με τις εξής στήλες:

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Σκεφτείτε ότι ο υπολογισμός του JOIN απαιτεί ένα διπλό for-loop
  - Δεν είναι ακριβές αυτό, αλλά θα σας βοηθήσει στην κατανόηση

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
3	Mary	3.5	1200	3	EMP	ECE	YES

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Inner Join (ή Join)

```
SELECT sName, cName, major
FROM    Student JOIN Apply ON
            (Student.sID = Apply.sID AND dec = 'YES')
```

- Μετά εκτελείται το SELECT

sName	cName	major
Jim	MIT	CS
Mary	EMP	ECE

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
3	Mary	3.5	1200	3	EMP	ECE	YES

Ισοδύναμα Ερωτήματα

```
SELECT sName, cName, major
FROM    Student JOIN Apply ON
            (Student.sID = Apply.sID AND dec = 'YES')
```

```
SELECT sName, cName, major
FROM    Student, Apply
WHERE Student.sID = Apply.sID AND dec = 'YES'
```

Το 2ο ερώτημα θα το κάνει το ΣΔΒΔ να εκτελεστεί σαν το 1ο

Διαφορά υπάρχει μόνο αν το JOIN που θέλουμε να εκτελεστεί είναι OUTER JOIN (θα το δούμε σε λίγο)

Σε OUTER JOIN ο 1ος τρόπος είναι ο σωστός

Ισοδύναμα Ερωτήματα

```
SELECT sName, cName, major
FROM    Student JOIN Apply ON
            (Student.sID = Apply.sID AND dec = 'YES')
```

```
SELECT sName, cName, major
FROM    Student JOIN Apply ON Student.sID = Apply.sID
WHERE  dec = 'YES'
```

Το 2ο ερώτημα θα το κάνει το ΣΔΒΔ να εκτελεστεί σαν το 1ο

Διαφορά υπάρχει μόνο αν το JOIN που θέλουμε να εκτελεστεί είναι OUTER JOIN (θα το δούμε σε λίγο)

Σε OUTER JOIN ο 1ος τρόπος είναι ο σωστός

Σημαντικές Λεπτομέρειες

```
SELECT sName, cName, major
FROM    Student JOIN Apply ON
            (Student.sID = Apply.sID AND dec = 'YES')
```

- Είναι απαραίτητο να δηλώνουμε τις συνθήκες με τις οποίες θα συνδυάζονται πλειάδες από τις 2 σχέσεις που κάνουμε JOIN
- Μπορούμε αν θέλουμε να δηλώσουμε και ψευδώνυμα στα ονόματα των σχέσεων (με, ή χωρίς τη χρήση του AS)

```
SELECT sName, cName, major
FROM    Student S JOIN Apply A ON
            (S.sID = A.sID AND dec = 'YES')
```

```
SELECT sName, cName, major
FROM    Student as S JOIN Apply as A ON
            (S.sID = A.sID AND dec = 'YES')
```

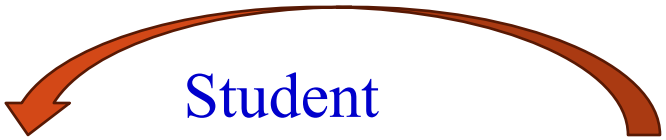
ΠΡΟΣΟΧΗ: Στο παραπάνω ερώτημα, μετά τη δήλωση ψευδώνυμων, στο υπόλοιπο ερώτημα υπάρχουν μόνο οι πίνακες S και A

- Θα πάρετε σφάλμα αν πάτε να αναφερθείτε στο Student.sID ή στο Apply.sID

Και αν δεν έχουμε συνθήκη Join;

```
SELECT *
FROM   Student, Apply
```

- Το παραπάνω ερώτημα λέγεται καρτεσιανό γινόμενο (Cross Product)
- Παράγει όλους τους συνδυασμούς πλειάδων από τις 2 σχέσεις
- Συνεπώς Tuples(Student) * Tuples(Apply) πλειάδες στο αποτέλεσμα



Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Natural Join

```
SELECT DISTINCT Student.sID, GPA
FROM    Student NATURAL JOIN Apply
```

- Το NATURAL JOIN επιβάλει ισότητα στα γνωρίσματα των 2 σχέσεων που έχουν το ίδιο όνομα
- Στην έξοδό του κρατάει ένα μόνο γνώρισμα από κάθε ζευγάρι με το ίδιο όνομα

sID	sName	GPA	hs	cName	major	dec

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Θυμηθείτε

```
SELECT *
FROM   Student JOIN Apply ON (Student.sID = Apply.sID)
```

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO
3	Mary	3.5	1200	3	EMP	ECE	YES

Μπορούν να εμφανιστούν και οι φοιτητές που ΔΕΝ έχουν κάνει αίτηση;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Left Outer Join

SELECT *

FROM Student Left Outer JOIN Apply ON Student.sID = Apply.sID

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO
2	Peter	3.7	800	NULL	NULL	NULL	NULL
3	Mary	3.5	1200	3	EMP	ECE	YES

Εμφανίζονται στο αποτέλεσμα ΟΛΑ τα tuples της ΑΡΙΣΤΕΡΗΣ πλευράς του JOIN. Όσα δεν κάνουν Join με τη δεξιά πλευρά, εμφανίζονται να κάνουν Join με NULL τιμές.

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Right Outer Join & Outer Join

Right Outer Join: Εμφανίζονται στο αποτέλεσμα ΌΛΑ τα tuples της ΔΕΞΙΑΣ πλευράς του JOIN. Όσα δεν κάνουν Join με την αριστερή πλευρά, εμφανίζονται να κάνουν Join με NULL τιμές.

Outer Join: Εμφανίζονται στο αποτέλεσμα ΌΛΑ τα tuples, τόσο της ΑΡΙΣΤΕΡΗΣ, όσο και της ΔΕΞΙΑΣ πλευράς του JOIN.

Όσα tuples δεν κάνουν Join με την απέναντι πλευρά, εμφανίζονται να κάνουν Join με NULL τιμές.

Outer Joins – Συνθήκες μόνο στο ON

Θυμηθείτε…

```
SELECT *
FROM    Student JOIN Apply ON
            (Student.sID = Apply.sID AND dec = 'YES')
```

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
3	Mary	3.5	1200	3	EMP	ECE	YES

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Left Outer Join

```
SELECT *
FROM    Student LEFT OUTER JOIN Apply ON
            (Student.sID = Apply.sID AND dec = 'YES')
```

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
2	Peter	3.7	800	NULL	NULL	NULL	NULL
3	Mary	3.5	1200	3	EMP	ECE	YES

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Left Outer Join (Λάθος αποτέλεσμα)

ΔΕΝ είναι ισοδύναμο το παρακάτω ερώτημα

```
SELECT *
FROM    Student LEFT OUTER JOIN Apply ON Student.sID = Apply.sID
WHERE  dec = 'YES'
```

1ο βήμα είναι το Left Outer Join στο FROM

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
1	Jim	3.0	800	1	TUC	CS	NO
2	Peter	3.7	800	NULL	NULL	NULL	NULL
3	Mary	3.5	1200	3	EMP	ECE	YES

2ο βήμα είναι το WHERE, που ικανοποιείται από 2 μόνο tuples

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.0	800	1	MIT	CS	YES
3	Mary	3.5	1200	3	EMP	ECE	YES

Διαφορετικό αποτέλεσμα από την προηγούμενη διαφάνεια!

Υπολογισμοί + Ψευδώνυμα σε SELECT

```
SELECT   sID, sName, 2*GPA, hs
FROM     Student
```

sID	sName	2*GPA	hs
1	Jim	7.0	1200
2	Peter	7.4	800
3	Mary	7.0	1200

```
SELECT   sID, sName as name, 2*GPA as newGPA, hs
FROM     Student
```

sID	name	newGPA	hs
1	Jim	7.0	1200
2	Peter	7.4	800
3	Mary	7.0	1200

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec

Join Τριών Σχέσεων

```
SELECT   Student.sID, sName, Apply.cName, enr
FROM     Student, Apply, College
WHERE    Student.sID = Apply.sID and
         Apply.cName = College.cName
```

Student.sID	sName	Apply.cName	Enr
1	Jim	MIT	22000
1	Jim	TUC	12000
3	Mary	EMP	17000

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Join Τριών Σχέσεων – Χρήση JOIN

```
SELECT    Student.sID, sName, Apply.cName, enr
FROM      Student JOIN Apply ON Student.sID = Apply.sID
            JOIN College ON (Apply.cName = College.cName)
```

Student.sID	sName	Apply.cName	enr
1	Jim	MIT	22000
1	Jim	TUC	12000
3	Mary	EMP	17000

Θυμηθείτε ότι ο τελεστής JOIN είναι δυαδικός τελεστής. Για να κάνουμε JOIN 3 πίνακες, χρειαζόμαστε 3-1 = 2 JOIN

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Μεταβλητές στο FROM

```
SELECT   S.sID, sName, A.cName, enr
FROM     Student S, Apply A, College C
WHERE    S.sID = A.sID and
         A.cName = C.cName
```

S.sID	sName	A.cName	enr
1	Jim	MIT	22000
1	Jim	TUC	12000
3	Mary	EMP	17000

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Self Join – Join του ίδιου πίνακα 2 φορές

- Πολλές φορές χρειάζεται να κάνουμε JOIN 2 αντίγραφα του ίδιου πίνακα. Γιατί;
- Θυμηθείτε ένα με SELECT-FROM-WHERE στον πίνακα Student μπορώ να επιλέξω Μαθητές που ικανοποιούν μία συνθήκη
- Πώς μπορώ να επιλέξω ζευγάρια μαθητών που ικανοποιούν μία συνθήκη;
  - Με το JOIN δημιουργούμε ζευγάρια από tuples 2 πινάκων
  - Εδώ θέλουμε ζευγάρια από tuples του πίνακα Student
  - Άρα θα χρειαστεί να κάνουμε JOIN τον πίνακα Student με τον εαυτό του (να κάνουμε JOIN 2 αντίγραφα του ίδιου πίνακα)
  - Εδώ είναι απαραίτητη η χρήση μεταβλητών στο FROM για να ξεχωρίζουμε τα 2 αντίγραφα της σχέσης Student

Self Join – Join του ίδιου πίνακα 2 φορές

Όταν στο FROM χρησιμοποιείται 2 φορές ο ίδιος πίνακας, είναι απαραίτητη η χρήση μεταβλητών

```
SELECT   S1.sID, s1.GPA, S2.sID, S2.GPA
FROM     Student S1, Student S2
WHERE    S1.GPA > S2.GPA
```

Επίσης σωστό: Student as S1

S1

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

S2

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Self Join – Join του ίδιου πίνακα 2 φορές

Όταν στο FROM χρησιμοποιείται 2 φορές ο ίδιος πίνακας, είναι απαραίτητη η χρήση μεταβλητών

```
SELECT   S1.sID, s1.GPA, S2.sID, S2.GPA
FROM     Student S1, Student S2
WHERE    S1.GPA > S2.GPA
```

Επίσης σωστό:
Student as S1

S1.sID	S1.GPA	S2.sID	S2.GPA
2	3.7	1	3.0
2	3.7	3	3.5
3	3.5	1	3.0

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

2ο Παράδειγμα Self Join

Βρες τα ζεύγη μαθητών με ίδιο hs (1η προσπάθεια)

```
SELECT   S1.sID, s1.hs, S2.sID, S2.hs
FROM     Student S1, Student S2
WHERE    S1.hs = S2.hs
```

S1.sID	S1.hs	S2.sID	S2.hs
1	800	1	800
1	800	2	800
2	800	1	800
2	800	2	800
3	1200	3	1200

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

2ο Παράδειγμα Self Join

Βρες τα ζεύγη μαθητών με ίδιο hs (2η προσπάθεια)

```
SELECT   S1.sID, s1.hs, S2.sID, S2.hs
FROM     Student S1, Student S2
WHERE    S1.hs = S2.hs and S1.sID <> S2.sID
```

S1.sID	S1.hs	S2.sID	S2.hs
1	800	2	800
2	800	1	800

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

2ο Παράδειγμα Self Join

Βρες τα ζεύγη μαθητών με ίδιο hs (3η προσπάθεια)

```
SELECT   S1.sID, s1.hs, S2.sID, S2.hs
FROM     Student S1, Student S2
WHERE    S1.hs = S2.hs and S1.sID < S2.sID
```

S1.sID	S1.hs	S2.sID	S2.hs
1	800	2	800

College

cName	state	enr
MIT	MASS	22000
TUC	Chania	12000
EMP	Attiki	17000

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Ένωση: UNION και UNION ALL

SQL QUERY_1

union

SQL QUERY_2

Τα 2 ερωτήματα πρέπει να είναι συμβατά ως προς τον αριθμό των γνωρισμάτων που επιστρέφουν και τον τύπο τους

Εκτελεί τα 2 ερωτήματα και επιστρέφει την ΕΝΩΣΗ των αποτελεσμάτων, αφαιρώντας τα διπλότυπα

SQL QUERY_1

union all

SQL QUERY_2

Τα 2 ερωτήματα πρέπει να είναι συμβατά ως προς τον αριθμό των γνωρισμάτων που επιστρέφουν και τον τύπο τους

Εκτελεί τα 2 ερωτήματα και επιστρέφει την ΕΝΩΣΗ των αποτελεσμάτων, χωρίς να αφαιρέσει τα διπλότυπα

Ένωση: UNION και UNION ALL

SELECT sName as name FROM Student

union

SELECT cName as name FROM College

(χωρίς διπλότυπα – **<u>συχνά</u>** ταξινομημένα αποτελέσματα λόγω αφαίρεσης διπλότυπων)

name
Jim
Peter
Mary
Jim

name
Mary
MIT
TUC

Αποτέλεσμα

name
Jim
Mary
MIT
Peter
TUC

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
TUC	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec

Ένωση: UNION και UNION ALL

```
SELECT  sName as name FROM  Student
union all
SELECT  cName as name FROM  College
```

(με διπλότυπα – όχι ταξινομημένα)

name
Jim
Peter
Mary
Jim
Mary
MIT
TUC

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
TUC	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec

Τομή: INTERSECT και INTERSECT ALL

SQL QUERY_1

intersect

SQL QUERY_2

Τα 2 ερωτήματα πρέπει να είναι συμβατά ως προς τον αριθμό των γνωρισμάτων που επιστρέφουν και τον τύπο τους

Επιστρέφει την ΤΟΜΗ (= tuples που επιστρέφονται και από τα 2 ερωτήματα) των αποτελεσμάτων, αφαιρώντας τα διπλότυπα.

SQL QUERY_1

intersect all

SQL QUERY_2

Τα 2 ερωτήματα πρέπει να είναι συμβατά ως προς τον αριθμό των γνωρισμάτων που επιστρέφουν και τον τύπο τους

Ένα tuple που εμφανίζεται X φορές στο 1ο αποτέλεσμα και Y φορές στο 2ο αποτέλεσμα επιστρέφεται min(X,Y) φορές

Τομή: INTERSECT

SELECT sName as name FROM Student

intersect

SELECT cName as name FROM College

(χωρίς διπλότυπα)

name
Jim
Peter
Mary
Jim

name
Mary
MIT
Jim

Αποτέλεσμα

name
Jim
Mary

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec

Τομή: INTERSECT

SELECT sName as name FROM Student

intersect

SELECT cName as name FROM College

(χωρίς διπλότυπα)

Ίδιο με

SELECT distinct sName as name

FROM Student, College

WHERE sName = cName

name
Jim
Mary

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec

Διαφορά: EXCEPT και EXCEPT ALL

SQL QUERY_1

except

SQL QUERY_2

Τα 2 ερωτήματα πρέπει να είναι συμβατά ως προς τον αριθμό των γνωρισμάτων που επιστρέφουν και τον τύπο τους

Επιστρέφει tuples που εμφανίζονται στο 1ο αποτέλεσμα και όχι στο 2ο αποτέλεσμα, αφαιρώντας τα διπλότυπα.

SQL QUERY_1

except all

SQL QUERY_2

Τα 2 ερωτήματα πρέπει να είναι συμβατά ως προς τον αριθμό των γνωρισμάτων που επιστρέφουν και τον τύπο τους

Ένα tuple που εμφανίζεται X φορές στο 1ο αποτέλεσμα και Y φορές στο 2ο αποτέλεσμα επιστρέφεται max(X-Y, 0) φορές

Διαφορά: EXCEPT

```
SELECT  sName as name FROM  Student
EXCEPT
SELECT  cName as name FROM  College
```

name
Jim
Peter
Mary
Jim

name
Mary
MIT
Jim

Αποτέλεσμα

name
Peter

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Διαφορά: EXCEPT

μαθητές που έκαναν αίτηση σε CS αλλά όχι σε EE

```
SELECT  sID FROM Apply WHERE major ='CS'
EXCEPT
SELECT  sID FROM Apply WHERE major ='EE'
```

sID
1
2

sID
2

Αποτέλεσμα

sID
1

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Διαφορά: EXCEPT ALL

SELECT sName as name FROM Student

EXCEPT ALL

SELECT cName as name FROM College

name
Jim
Peter
Mary
Jim

name
Mary
MIT
Jim

Αποτέλεσμα

name
Jim
Peter

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Συναθροιστικές Τιμές

- Μπορούμε να υπολογίσουμε συναθροιστικές τιμές πάνω σε ολόκληρο το αποτέλεσμα που προκύπτει από το FROM-WHERE

- Μπορούμε επίσης να χωρίσουμε τα tuples που προκύπτουν από το FROM-WHERE σε ομάδες και μετά να υπολογίσουμε συναθροιστικές συναρτήσεις **ΑΝΑ ομάδα** (εντολή GROUP BY)

Βασικές Συναθροιστικές Συναρτήσεις

- count(attribute): Μετράει πόσες εγγραφές δεν έχουν NULL τιμές στο συγκεκριμένο attribute
- count(distinct attribute): Μετράει πόσες διακριτές εγγραφές στο συγκεκριμένο attribute δεν έχουν NULL τιμές
- count(*): Μετράει πόσες εγγραφές υπάρχουν στο αποτέλεσμα ή στην ομάδα

- max(attribute): Μέγιστη τιμή του attribute (αγνοεί τα NULL)
- min(attribute): Ελάχιστη τιμή του attribute (αγνοεί τα NULL)

- sum(attribute): Άθροισμα τιμών του attribute (αγνοεί τα NULL)
- avg(attribute): Μέσος όρος τιμών του attribute (αγνοεί τα NULL)

AGGREGATES ΧΩΡΙΣ GROUP BY

Ποιος είναι ο μέγιστος μέσος όρος μαθητή που έχει κάνει αίτηση στο MIT;

```
SELECT   MAX(GPA) as koryfaios
FROM     Student, Apply
WHERE    Student.sID = Apply.sID and cName = 'MIT'
```

Αν βάλουμε SELECT *, θα δούμε πάνω σε ποιες πλειάδες υπολογίζεται η συναθροιστική τιμή

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

AGGREGATES ΧΩΡΙΣ GROUP BY

```
SELECT    MAX(GPA) as koryfaios
FROM      Student, Apply
WHERE     Student.sID = Apply.sID and cName = 'MIT'
```

koryfaios
3.6

Μετά το FROM και το WHERE έχει μείνει το εξής:

Student.sID	sName	GPA	hs	Apply.sID	cName	major	dec
1	Jim	3.5	1100	1	MIT	MPD	NO
3	Mary	3.6	1200	3	MIT	EE	NO

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Αριθμός Πλειάδων στο Αποτέλεσμα

COUNT(*) για επιστροφή αριθμού πλειάδων

COUNT(DISTINCT attributeName) για επιστροφή αριθμού πλειάδων με διαφορετικό attributeName

- Μόνο 1 attributeName στο DISTINCT

num	num2
3	2

```
SELECT count(*) as num, count(distinct Student.sID)  as num2
FROM   Student, Apply
WHERE Student.sID = Apply.sID and cName = 'MIT'
```

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

DISTINCT Τιμές

SELECT DISTINCT sName
FROM STUDENT

sName
Jim
Peter
NULL
Mac

Το NULL επιστρέφεται ως distinct τιμή

SELECT COUNT(sName) as cnt1,
COUNT(DISTINCT sName) as cnt2
FROM STUDENT

cnt1	cnt2
4	3

Το NULL δεν μετριέται στο count ως distinct τιμή!

Το COUNT(sName) μετράει τα non null ονόματα

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	NULL	3.6	1200
4	Jim	NULL	NULL
5	Mac	NULL	900

Αριθμός Πλειάδων στο Αποτέλεσμα

Αν θέλαμε COUNT(DISTINCT) σε πολλά attributes, τότε πρέπει να χρησιμοποιήσουμε nested subquery (θα το δούμε αργότερα)

```
SELECT count(*)
FROM    (SELECT DISTINCT attr1, attr2, …, attrn
          FROM R1, …., Rm
          WHERE… ) X
```

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

GROUP BY

Επιτρέπει δημιουργία ομάδων και υπολογισμού συναθροιστικών τιμών ΑΝΑ ΟΜΑΔΑ

Βρείτε για κάθε κολλέγιο τον αριθμό των αιτήσεων και τον αριθμό των μαθητών που έχουν κάνει αίτηση

```
SELECT cName, count(*) as num,
              count(distinct sID)  as num2
FROM   Apply
GROUP BY  cName
```

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

GROUP BY – Πώς Εκτελείται;

```
SELECT cName, count(*) as num, count(distinct sID)  as num2
FROM  Apply
GROUP BY  cName
```

1. Εκτελείται το FROM (εδώ δε γίνεται τίποτα – μόνο ένας πίνακας)
2. Εκτελείται το WHERE για να φιλτραριστούν οι εγγραφές που προκύπτουν από το FROM (εδώ δεν υπάρχει WHERE)
3. Ομαδοποιούνται τα tuples που μένουν μετά το WHERE σύμφωνα με τα γνωρίσματα ομαδοποίησης

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

GROUP BY – Πώς Εκτελείται;

```
SELECT cName, count(*) as num, count(distinct sID)  as num2
FROM  Apply
GROUP BY  cName
```

Μετά το WHERE

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Μετά το GROUP BY

sID	cName	major	dec	
1	MIT	MPD	NO	count(*) = 3 count(distinct sID) = 2
1	MIT	CS	NO	
3	MIT	EE	NO	
1	TUC	CS	YES	count(*) = 2 count(distinct sID) = 2
4	TUC	CS	YES	
5	EMP	CS	NO	count(*) = 1 count(distinct sID) = 1

GROUP BY – Πώς Εκτελείται;

```
SELECT cName, count(*) as num, count(distinct sID)  as num2
FROM   Apply
GROUP BY  cName
```

Αποτέλεσμα

cName	num	num2
TUC	2	2
MIT	3	2
EMP	1	1

Μετά το GROUP BY

sID	cName	major	dec	
1	MIT	MPD	NO	count(*) = 3 count(distinct sID) = 2
1	MIT	CS	NO	
3	MIT	EE	NO	
1	TUC	CS	YES	count(*) = 2 count(distinct sID) = 2
4	TUC	CS	YES	
5	EMP	CS	NO	count(*) = 1 count(distinct sID) = 1

GROUP BY – 2ο Παράδειγμα

Βρείτε για κάθε κολλέγιο το μέγιστο και τον ελάχιστο μέσο όρο μαθητή που έχει κάνει αίτηση σε αυτό

```
SELECT      cName, max(GPA) as mx, min(GPA) as mn
FROM        Apply, Student
WHERE       Apply.sID = Student.sID
GROUP BY    cName
```

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

GROUP BY – Πώς Εκτελείται;

```
SELECT      cName, max(GPA) as mx, min(GPA) as mn
FROM        Apply, Student
WHERE       Apply.sID = Student.sID
GROUP BY    cName
```

Μετά το WHERE

sID	sName	GPA	hs	sID	cName	major	dec
1	Jim	3.5	1100	1	TUC	CS	YES
1	Jim	3.5	1100	1	MIT	MPD	NO
1	Jim	3.5	1100	1	MIT	CS	NO
3	Mary	3.6	1200	3	MIT	EE	NO
4	Jim	3.8	1400	4	TUC	CS	YES
5	Mac	3.4	900	5	EMP	CS	NO

GROUP BY – Πώς Εκτελείται;

```
SELECT    cName, max(GPA) as mx, min(GPA) as mn
FROM      Apply, Student
WHERE     Apply.sID = Student.sID
GROUP BY  cName
```

Αποτέλεσμα

cName	mx	mn
TUC	3.8	3.5
MIT	3.6	3.5
EMP	3.4	3.4

Μετά το GROUP BY

sID	sName	GPA	hs	sID	cName	major	dec
1	Jim	3.5	1100	1	TUC	CS	YES
4	Jim	3.8	1400	4	TUC	CS	YES
1	Jim	3.5	1100	1	MIT	MPD	NO
1	Jim	3.5	1100	1	MIT	CS	NO
3	Mary	3.6	1200	3	MIT	EE	NO
5	Mac	3.4	900	5	EMP	CS	NO

max(GPA) = 3.8
min(GPA) = 3.5

max(GPA) = 3.6
min(GPA) = 3.5

max(GPA) = 3.4
min(GPA) = 3.4

Βασικοί Κανόνες

- Στο SELECT εμφανίζονται είτε γνωρίσματα, είτε συναθροιστικές τιμές (sum, count, max, min, avg) πάνω σε κάποια attributes
- Στο WHERE οι συνθήκες εφαρμόζονται πάνω σε πλειάδες
- Αν υπάρχει GROUP BY, ό,τι εμφανίζεται στο SELECT **πρέπει** να είναι είτε γνωρίσματα ομαδοποίησης είτε συναθροιστική τιμή
- Αν ΔΕΝ υπάρχει GROUP BY οι συναθροιστικές τιμές υπολογίζονται πάνω σε όλες τις πλειάδες του αποτελέσματος
- Συναρτήσεις συνάθροισης ΔΕΝ επιτρέπονται στο WHERE

Σωστό Συντακτικά ή Όχι ;;;

```
SELECT      cName, max(GPA) as mx, min(GPA) as mn
FROM        Apply, Student
WHERE       Apply.sID = Student.sID
GROUP BY    cName, major
```

Σωστό: Στο SELECT εμφανίζονται είτε γνωρίσματα που υπάρχουν στο GROUP BY, είτε συναθροιστικές συναρτήσεις

Σωστό Συντακτικά ή Όχι ;;;

```
SELECT      cName, major, max(GPA) as mx, min(GPA) as mn
FROM        Apply, Student
WHERE       Apply.sID = Student.sID
GROUP BY    cName
```

Λάθος: Στο SELECT εμφανίζεται το major που δεν εμφανίζεται στο GROUP BY

Γιατί όμως είναι λάθος;

- Σκεφτείτε ότι μέσα στο group ενός κολλεγίου μπορεί να υπάρχουν αιτήσεις για πολλά major (τμήματα)
- Ποιο από αυτά πρέπει να τυπωθεί για το group; Καμία σωστή απάντηση…
- Για αυτό είναι λάθος …

COUNT + Outer Joins

Για κάθε κολλέγιο, εμφάνισε τον αριθμό των αιτήσεων που έχουν γίνει σε αυτό. Εκτυπώστε πληροφορία και για κολλέγια στα οποία δεν έχει γίνει καμία αίτηση.

Σημείωση: Η τελευταία πρόταση πρέπει να σας παραπέμπει σε outer join

```
SELECT      College.cName, count(major) as applications
FROM        College left outer join Apply ON
                (College.cName = Apply.cName)
GROUP BY    College.cName
```

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

COUNT + Outer Joins: Τι προσέχουμε

```
SELECT      College.cName, count(major) as applications
FROM        College left outer join Apply ON
                (College.cName = Apply.cName)
GROUP BY    College.cName
```

1. Χρησιμοποίησα left outer join με την College αριστερά για να βεβαιωθώ ότι κάθε Κολλέγιο θα εμφανιστεί στο αποτέλεσμα.
2. Left outer join => χρειάζομαι τη συνθήκη του JOIN στο ON (και όχι στο WHERE)
3. Στο GROUP BY πρέπει να έχω γνωρίσματα της σχέσης στα αριστερά στο left outer join (μη χρησιμοποιήσετε εδώ Apply.cName)
4. Στο count(major) φρόντισα να βάλω ένα attribute της δεξιάς σχέσης που δε μπορεί κανονικά να πάρει NULL τιμές
   - Κομμάτια του κλειδιού της δεξιάς σχέσης είναι πάντα καλή επιλογή

COUNT + Outer Joins: Τι προσέχουμε

Στο count(major) φρόντισα να βάλω ένα attribute της δεξιάς σχέσης που δε μπορεί κανονικά να πάρει NULL τιμές

- Κομμάτια του κλειδιού της δεξιάς σχέσης είναι πάντα καλή επιλογή

ΓΙΑΤΙ;;;;;;;

- Για κολλέγια για τα οποία υπάρχουν αιτήσεις, το major δε θα είναι NULL. Άρα το count(major) θα μου επιστρέφει τον αριθμό των πλειάδων (αιτήσεων) για κάθε group (κολλέγιο) του GROUP BY

College left outer join Apply ON College.cName = Apply.cName

College.cName	state	enr	sID	Apply.cName	major	dec
Mary	MD	1000	1	Mary	CS	YES
Mary	MD	1000	2	Mary	CS	YES
MIT	MA	17000	1	MIT	MPD	NO
MIT	MA	17000	2	MIT	EE	NO
Jim	CH	12000	NULL	NULL	NULL	NULL

COUNT + Outer Joins: Τι προσέχουμε

College left outer join Apply ON College.cName = Apply.cName

College.cName	state	enr	sID	Apply.cName	major	dec
Mary	MD	1000	1	Mary	CS	YES
Mary	MD	1000	2	Mary	CS	YES
MIT	MA	17000	1	MIT	MPD	NO
MIT	MA	17000	2	MIT	EE	NO
Jim	CH	12000	NULL	NULL	NULL	NULL

- Για κολλέγια για τα οποία ΔΕΝ υπάρχουν αιτήσεις, αυτά θα εμφανίζονται να κάνουν join με NULL τιμές δεξιά
  - Το group τους θα έχει 1 tuple
  - Το major σε αυτό το 1 tuple θα είναι NULL
  - Το count(major) σε αυτό το group θα επιστρέψει 0.
- Προσέξτε ότι θα ήταν λάθος να βάλω count(*) αντί για count σε ένα attribute της δεξιάς σχέσης, γιατί το count(*) θα επιστρέψει 1 για κολλέγια με 0 αιτήσεις

Εμφάνιση Ορισμένων μόνο Group

- Στο WHERE ορίζουμε συνθήκες
  - Οι συνθήκες αυτές περιορίζουν το ποια tuples που έρχονται από το FROM θα εμφανιστούν στο αποτέλεσμα
- Μετά είπαμε ότι μπορούμε να ομαδοποιήσουμε τα tuples με τη χρήση του GROUP BY
- Μπορούμε να επιστρέψουμε πληροφορία μόνο για κάποια groups που ικανοποιούν κάποια συνθήκη;
  - Π.χ., περιέχουν αρκετά tuples

- HAVING
  - Μπαίνει μετά το GROUP BY (και πριν από ORDER BY, LAST)
  - Έχει νόημα η συνθήκη να είναι πάνω σε **συναθροιστικές τιμές** του group
    - **Συναθροιστικές τιμές μπαίνουν ΜΟΝΟ στο SELECT και στο HAVING**

GROUP BY και HAVING

Γενική Δομή (θα εμπλουτιστεί μετά)

5 SELECT A1, A2, …, An
1 FROM R1, R2, …, Rm
2 WHERE conditions
3 GROUP BY list of attribute names
4 HAVING condition on group aggr values
6 ORDER BY Ai, Aj, … ASC/DESC
7 LIMIT K

Πολλά μέρη του ερωτήματος είναι προαιρετικά.
Η σειρά εκτέλεσης τους φαίνεται από τα νούμερα στα αριστερά

HAVING

Το HAVING μας επιτρέπει να ελέγχουμε συνθήκες ΑΝΑ group

Βρείτε τα κολλέγια με τουλάχιστον 3 αιτήσεις

```
SELECT   cName
FROM     Apply
GROUP BY  cName
HAVING count(*) >= 3
```

cName
MIT

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

HAVING

Βρείτε τα αναγνωριστικά των μαθητών που έχουν τουλάχιστον 2 αιτήσεις που έχουν απορριφθεί

```
SELECT   sID
FROM     Apply
WHERE    dec='NO'
GROUP BY  sID
HAVING count(*) >= 2
```

sID
1

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

HAVING

Βρείτε τα αναγνωριστικά των μαθητών που έχουν **το πολύ** 1 αίτηση που έχει απορριφθεί

Αυτό είναι σημαντικά πιο δύσκολο ερώτημα. Γιατί;

- Μπορεί κάποιος να έχει κάνει αίτηση και να έχει 0-1 'NO' στο dec
- Μπορεί κάποιος να έχει μην έχει κάνει καν αίτηση. Δεν υπάρχει τότε στον πίνακα Apply (αλλά μόνο στο Student)

Θα δούμε 2 σωστές προσεγγίσεις:

- Με left outer join
- Με EXCEPT

HAVING – Λάθος προσέγγιση

Βρείτε τα αναγνωριστικά των μαθητών που έχουν **το πολύ** 1 αίτηση που έχει απορριφθεί

```
SELECT   sID
FROM     Apply
WHERE    dec='NO'
GROUP BY  sID
HAVING count(*) <= 1
```

sID
3
5

Γιατί είναι λάθος;

- Δεν εμφανίζει όσους δεν έχουν NO σε κάποια αίτηση
- Δεν εμφανίζει όσους δεν έχουν κάνει αίτηση

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

HAVING – Σωστή προσέγγιση 1

Βρείτε τα αναγνωριστικά των μαθητών που έχουν **το πολύ** 1 αίτηση που έχει απορριφθεί

```
SELECT   Student.sID as id
FROM     Student left outer join Apply
             ON Student.sID = Apply.sID AND dec='NO'
GROUP BY  Student.sID
HAVING count(cName) <= 1
```

id
2
3
4
5

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

HAVING – Σωστή προσέγγιση 2

Λύση: Αφαίρεσε από όλους τους μαθητές, αυτούς
που έχουν 2 ή περισσότερα 'NO' (δείτε 4 διαφάνειες πριν)

```
SELECT   sID from Student
EXCEPT
SELECT   sID
FROM     Apply
WHERE    dec='NO'
GROUP BY  sID
HAVING count(*) >= 2
```

sID
2
3
4
5

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Επιπλέον Λεπτομέρειες

- Ποιες συνθήκες είναι εύκολο να ελεγχθούν σε ένα JOIN;
- Ποιες συνθήκες είναι δύσκολο να ελεγχθούν σε ένα JOIN;
- Μπορούμε να κάνουμε NATURAL JOIN με τη χρήση του JOIN τελεστή;
  - Λέγοντας ρητά σε ποια attributes θέλουμε να εφαρμοστεί η ισότητα;

Θυμηθείτε …

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec = 'YES')
```

- Είναι εύκολο να βρούμε μαθητές που έχουν τουλάχιστον 1 θετική απάντηση σε αίτηση
  - Αρκεί να βρούμε ένα συνδυασμό tuples για τον οποίον, dec='YES'
  - Μαθητής με πολλά 'YES' θα βρεθεί πολλές φορές, καθώς εξετάζουμε διαφορετικούς συνδυασμούς

Ισχύει ότι Student.sID = Apply.sID AND dec = 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Επίσης Εύκολο …

```
SELECT sName, cName, major
FROM   Student JOIN Apply ON
           (Student.sID = Apply.sID AND dec != 'YES')
```

- Είναι εύκολο να βρούμε μαθητές που τουλάχιστον 1 φορά δεν ικανοποιούν κάποια συνθήκη
  - Αρκεί να βρούμε ένα συνδυασμό tuples για τον οποίον δεν ικανοποιείται η συνθήκη

Ισχύει ότι Student.sID = Apply.sID AND dec != 'YES' ;

Student

sID	sName	GPA	hs
1	Jim	3.0	800
2	Peter	3.7	800
3	Mary	3.5	1200

Apply

sID	cName	major	dec
1	MIT	CS	YES
1	TUC	CS	NO
3	EMP	ECE	YES

Δύσκολη Συνθήκη

- Βρες τους μαθητές που δεν έχουν **καμία αίτηση** που να έχει απορριφθεί
  - Θυμηθείτε: Κοιτάμε 1-1 τους συνδυασμούς tuples των 2 σχέσεων
  - Αν σε ένα συνδυασμό βρούμε μία αίτηση που να είναι δεκτή, πώς ξέρουμε εκείνη τη στιγμή αν **για όλους** τους συνδυασμούς αυτού του μαθητή, ότι επίσης dec='YES';

    ΔΕΝ ΤΟ ΞΕΡΟΥΜΕ
  - Όταν κοιτάμε 1 συνδυασμό από tuples, εκείνη την ώρα **δεν έχουμε γνώση για το τι συμβαίνει στους άλλους συνδυασμούς**

- Πιθανές λύσεις:
  - EXCEPT (παρόμοια με αυτή που είδαμε πριν λίγες διαφάνειες)
  - LEFT OUTER JOIN χωρίς HAVING
  - LEFT OUTER JOIN με HAVING
  - NESTED SUBQUERIES (Θα τα δούμε στην επόμενη ενότητα διαφανειών)

- ΣΗΜΕΙΩΣΗ: Επίσης δύσκολη συνθήκη είναι να βρούμε μαθητές που ΠΑΝΤΑ ικανοποιούν μία συνθήκη (το EXCEPT είναι η εύκολη λύση για τέτοια ερωτήματα)

Λύση με EXCEPT

Λύση: Αφαίρεσε από όλους τους μαθητές αυτούς που έχουν 1 ή περισσότερα 'NO'

```
SELECT   sID from Student
EXCEPT
SELECT   sID
FROM     Apply
WHERE    dec='NO'
```

sID
2
3
4
5

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
1	MIT	CS	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

LEFT OUTER JOIN χωρίς HAVING

Λύση 2: Αν προσπαθήσω να βρω το ποιοι έχουν αίτηση με dec='NO', όσοι μαθητές θέλω να επιστρέψω δε θα βρουν τέτοια αίτηση στην Apply για να συνδυαστούν μαζί της.

Συνεπώς, σε LEFT OUTER JOIN θα εμφανιστούν να κάνουν JOIN με NULL τιμές στα γνωρίσματα της Apply

```
SELECT   Student.sID
FROM     Student left outer join Apply
             ON Student.sID = Apply.sID AND dec='NO'
WHERE   Apply.cName is NULL
```

LEFT OUTER JOIN με HAVING

Λύση 2: Αν προσπαθήσω να βρω το ποιοι έχουν αίτηση με dec='NO', όσοι μαθητές θέλω να επιστρέψω δε θα βρουν τέτοια αίτηση για να συνδυαστούν.

Συνεπώς, σε LEFT OUTER JOIN θα εμφανιστούν να κάνουν JOIN με NULL

Συνεπώς, το COUNT σε ένα attribute της δεξιάς σχέσης που δεν επιτρέπει (κανονικά) NULL θα επιστρέψει 0.

```
SELECT   Student.sID
FROM     Student left outer join Apply
                 ON Student.sID = Apply.sID AND dec='NO'
GROUP BY  Student.sID
HAVING count(cName) = 0
```

Γιατί το Παρακάτω ΔΕΝ Είναι Σωστό;

```
SELECT   Student.sID
FROM     Student join Apply
             ON Student.sID = Apply.sID AND dec='NO'
GROUP BY  Student.sID
HAVING count(cName) = 0
```

- Μαθητές που δεν έχουν κάνει αίτηση, δεν κάνουν JOIN
  - Δεν υπάρχει tuple στην Apply με Student.sID = Apply.sID
- Μαθητές που έχουν μόνο 'YES' σε αιτήσεις τους, δεν κάνουν JOIN
  - Δεν υπάρχει joined tuple (στο sID) στην Apply με dec='NO'
- Και στις 2 περιπτώσεις έχουμε 0 tuples για αυτούς τους μαθητές, και δε δημιουργείται GROUP στο GROUP BY
- Κάντε σύγκριση με τη περίπτωση του LEFT OUTER JOIN

ΧΡΗΣΗ ΤΟΥ JOIN ΩΣ NATURAL JOIN (USING)

Το INNER JOIN μπορεί να λειτουργήσει ως NATURAL JOIN σε συνδυασμό με τον τελεστή USING

Το USING μας λέει σε ποια attributes να εφαρμόσουμε την ισότητα

```
SELECT   sID, sName, cName, enr
FROM     (Student  S join Apply A USING(sID))
         join College C USING(cName)
```

Χρήσιμο αν ΔΕ θέλουμε να εφαρμόσουμε στη συνένωση την ισότητα μεταξύ κάποιων γνωρισμάτων που τυχαίνουν να έχουν το ίδιο όνομα

Σκεφτείτε να έχετε γράψει το ερώτημα και μετά να μετονομάσετε ένα attribute σε ένα όνομα που περιέχει ο άλλος πίνακας

ΧΡΗΣΗ ΤΟΥ JOIN ΩΣ NATURAL JOIN (USING)

Το USING μας λέει σε ποια attributes να εφαρμόσουμε την ισότητα

Μέσα στην παρένθεση τα attributes χωρίζονται με κόμμα

ΔΕ συνδυάζεται με το ON – χρειάζεται χρήση του WHERE για συνένωση σε επιπλέον attributes

```
SELECT    sID, sName, cName
FROM      Student  S join Apply A USING(sID)
WHERE     sName < cName
```

OUTER JOINs ΩΣ NATURAL JOIN (USING)

Βρείτε για κάθε μαθητή τα ονόματα των κολλεγίων και τμημάτων στα οποία έχει κάνει αίτηση

```
SELECT    sID, sName, cName, major
FROM      Student  S join Apply A USING(sID)
```

Το παραπάνω ερώτημα ΔΕ θα εμφανίσει αποτελέσματα για μαθητές που ΔΕΝ έχουν κάνει αίτηση. Αν θέλουμε να τους συμπεριλάβουμε, χρειαζόμαστε LEFT OUTER JOIN

```
SELECT    sID, sName, cName, major
FROM      Student  S left outer join Apply A USING(sID)
```

Ερώτημα Εξάσκησης 1

Τυπώστε όλα τα στοιχεία των Κολλεγίων στα οποία έχουν γίνει δεκτοί μαθητές

```
SELECT DISTINCT College.cName, state, enr
FROM    College JOIN Apply ON
                College.cName = Apply.cName AND dec = 'YES'
```

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Ερώτημα Εξάσκησης 2

Τυπώστε ζευγάρια κολλεγίων που βρίσκονται στην ίδια πολιτεία (state)

```
SELECT C1.cName, C2.cName
FROM   College C1, College C2
WHERE C1.state = C2.state AND C1.cName < C2.cName
```

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Ερώτημα Εξάσκησης 3

Τυπώστε όλα τα στοιχεία των Κολλεγίων στα οποία έχουν κάνει αίτηση τουλάχιστον 2 μαθητές

```
SELECT College.cName, state, enr
FROM    College JOIN Apply ON College.cName = Apply.cName
GROUP BY College.cName
HAVING        count(*) >= 2
```

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Ερώτημα Εξάσκησης 4

Τυπώστε για κάθε Κολλέγιο το όνομά του και τον αριθμό των αιτήσεων σε αυτό. Να συμπεριλάβετε Κολλέγια με 0 αιτήσεις

```
SELECT College.cName, count(sID)
FROM    College LEFT OUTER JOIN Apply
                    ON College.cName = Apply.cName
GROUP BY College.cName
```

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Ερώτημα Εξάσκησης 5

Τυπώστε όλα τα στοιχεία των κολλεγίων στα οποία δεν έχει κάνει αίτηση κανένας μαθητής με μέσο όρο μεγαλύτερο του 3.65

Εύκολη λύση: Αφαίρεσε από όλα τα κολλέγια, αυτά στα οποία έχει κάνει αίτηση έστω ένας φοιτητής με μέσο όρο μεγαλύτερο του 3.65

```
SELECT * FROM College
EXCEPT
SELECT College.cName, state, enr
FROM    College JOIN Apply ON College.cName = Apply.cName
            JOIN Student ON Apply.sID = Student.sID AND GPA > 3.65
```

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Ερώτημα Εξάσκησης 5

Τυπώστε όλα τα στοιχεία των κολλεγίων στα οποία δεν έχει κάνει αίτηση κανένας μαθητής με μέσο όρο μεγαλύτερο του 3.65

```
SELECT College.cName, state, enr
FROM    College LEFT OUTER JOIN Apply
                ON College.cName = Apply.cName
                LEFT OUTER JOIN Student
                        ON Apply.sID = Student.sID AND GPA > 3.65
WHERE       Student.sID is NULL
```

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Ερώτημα Εξάσκησης 6

Τυπώστε τα sID των μαθητών που έχουν κάνει αίτηση σε ΌΛΑ τα Κολλέγια

ΔΥΣΚΟΛΟ ερώτημα (ονομάζεται ΔΙΑΙΡΕΣΗ ο συγκεκριμένος τελεστής)

Πώς θα το απαντήσουμε;

Αν κάποιος μαθητής έχει κάνει αίτηση σε όλα τα κολλέγια, πρέπει να υπολογίσουμε:

- Σε πόσα διακριτά κολλέγια έχει κάνει αίτηση ο κάθε μαθητής
- Να ελέγξουμε αν αυτός ο αριθμός είναι ίσος με τον αριθμό των Κολλεγίων
- Συνθήκη σε συναθροιστική τιμή => GROUP BY + HAVING
- Ποια συνάρτηση υπολογίζει τον αριθμό των διακριτών κολλεγίων;
  - count(distinct cName)
- Πώς υπολογίζουμε τον αριθμό των Κολλεγίων;
  - SELECT COUNT(*) FROM College

Ερώτημα Εξάσκησης 6

Τυπώστε τα sID των μαθητών που έχουν κάνει αίτηση σε ΌΛΑ τα Κολλέγια

```
SELECT      sID
FROM        Apply
GROUP BY    sID
HAVING      count(distinct cName) = (SELECT COUNT(*)
                                     FROM College)
```

Ομαδοποιεί τις αιτήσεις ανά μαθητή

Μετράει τα διαφορετικά κολλέγια στα οποία έχει κάνει αίτηση ο κάθε μαθητής

Μετράει τον αριθμό των Κολλεγίων

Οι παρενθέσεις γύρω από το SELECT ερώτημα στο HAVING είναι υποχρεωτικές